УДК 004.056.53

И. Н. Изотов
Научный руководитель: О. А. Пономарева
Уральский федеральный университет, Екатеринбург

БЕЗОПАСНОСТЬ ВЕБ-САЙТОВ: SQL-ИНЪЕКЦИЯ

Аннотация. Данная работа представляет собой обзор существующих методов определения SQL-инъекций, рассмотрение причин возникновения. На основе рассмотренного материала предлагаются методы по защите информаци на уровне ввода данных в текстовые формы.

Ключевые слова: информация; безопасность; SQL; SQL-инъекция; защита информации.

Веб-сайты обращаются к структурированной информации, которая размещена в базах данных. Пользователи взаимодействуют с данными через формы для ввода и отображения информации. Например, при аутентификации вводится логин и пароль, критерии проверяются на сервере. В базе данных находится информация о пользователях и определен уровень доступа. Иногда разработчики сайта не учитывают ситуацию, что в поле ввода информации, возможно, помещен текст, который нарушит работу сервера. Сервер обрабатывает полученные данные как заданную команду. Этой уязвимостью может воспользоваться злоумышленник, например, получив аутентификационные данные пользователя с правами администратора, — такая проблема получила название SQL-инъекции. После ввода логина и пароля, данные из текстовой формы передаются на сервер и подставляются в запрос, представленный следующим образом [1]:

select * from Users where login='username' AND password= 'pass',

где *username* и *pass* — введенный текст в поле данных на сайте.

Приложение авторизует пользователя в том случае, если пара логин и пароль будут определены в базе пользователей. Но, заменив *username* на фразу "*admin*' --", итоговый запрос подменит начальную команду:

select * from Users where login='***admin*\' -- '** AND password= 'pass'.

И при наличии пользователя с именем *admin* сайт позволит работать под его учетной записью без проверки пароля. База проигнорирует всю оставшуюся команду, стоящую после двух черточек "--", потому что в синтаксисе SQL так

обозначается комментарий. Символ “;” отделяет одну команду SQL от другой, поэтому возможно выполнение нескольких команд.

Рассмотрим варианты некорректных запросов, которые вызывают уязвимость к SQL-инъекциям в работе базы данных.

Причины уязвимости

Такие уязвимости возникают по следующим причинам [2, с. 3–4]:

– Динамическое построение SQL-запросов. Самый простой вариант для программиста — подставлять без дополнительной фильтрации введенные пользователем данные в готовую конструкцию запроса, к примеру:

select * from users where login =’request.getParameter(“name”)‘.

– Некорректная обработка исключений. При попытке отправки SQL-команды, содержащей ошибку, SQL-сервер отправит обратно пользователю информацию о ней. Злоумышленник, ориентируясь по этим сообщениям, вводит правильную команду.
– Некорректная обработка специальных символов. В базе данных MySQL, например, комментарий задается с помощью символов “--” или “#” [3]. Потому приложение должно обрабатывать такие символы.
– Некорректная обработка типов. При обработке символа апострофа (‘) закрывается уязвимость текстовых данных. Однако в числовых типах проблема остается. У злоумышленника есть возможность добавить команду, например:

select * from money where sum =
1 ***union select 1, username, password from users***.

С помощью *union* к исходному запросу присоединится еще одна таблица, в данном случае с названием *users*.

– Небезопасная конфигурация СУБД. Учетная запись сервера приложений СУБД по умолчанию обладает излишними правами, из-за чего база данных позволит злоумышленнику выполнять непредусмотренные команды.

С точки зрения программирования, задача защиты сервера от SQL-инъекций решаема. Администратор должен помнить об этой уязвимости и предпринять правильные меры по защите.

Защита

SQL-инъекция является распространенной и разрушающей уязвимостью, поэтому важно защитить приложение и сервер от данных атак [4, с. 149–152].

Прежде всего необходима фильтрация поступающих данных. Фильтрация происходит на сервере, так как из приложения возможен перехват и изменение информации.

Специальные символы текстовой информации, например, апостроф (‘) или косая черта (/), должны дополняться символом косой черты (/). Например, если в поле *username* будет значение «Д’Артаньян», то в базу данных будет записано значение «Д/’Артаньян», и ошибок не возникнет (рис. 1).

Для проверки численной информации пользуются проверкой типа. Сервер вернет корректную ошибку пользователю, который в числовое поле поместит текст (рис. 2).

В статье приводится листинг программы по проверке корректности ввода данных (рис. 3).

В приложении рекомендуется ограничивать количество символов для ввода информации, а на сервере — проверять ее. При переполнении поля запрос отклоняется.

Рекомендуется также отключить вывод ошибок от SQL-сервера, тогда злоумышленнику придется «вслепую» угадывать команды. Рекомендуется произвести настройку прав пользователей базы данных, с ограничением прав учетной записи сервера.

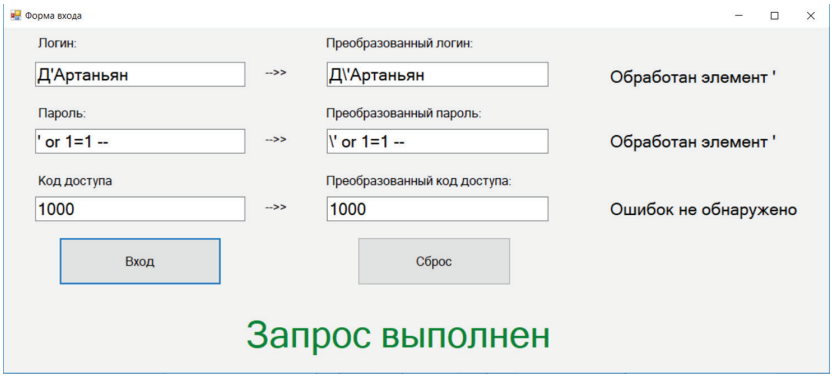

Рис. 1. Проверка специальных символов

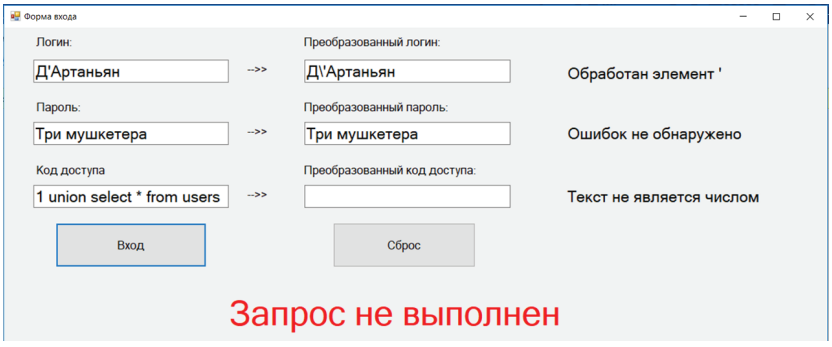

Рис. 2. Неудачная попытка доступа через числовое поле

```
private bool CheckOfString(TextBox TB, TextBox TB2, Label L, string type)
{
    TB2.Text = "";
    bool flag = true;
    if (TB.Text.Length == 0) { L.Text = "Пустая строка, запрос не выполнен";
        return false; }
    if (type == "text") {
        for (int i = 0; i < TB.Text.Length; i++) {
            if (TB.Text[i] == '\'')
            { TB2.Text += "\\\'";
                flag = false;
                L.Text = "Обработан элемент \'"; }
            else if (TB.Text[i] == '\"') {
                TB2.Text += "\\\"";
                flag = false;
                L.Text = "Обработан элемент \""; }
            else if (TB.Text[i] == '\\'){
                TB2.Text += "\\\\";
                flag = false;
                L.Text = "Обработан элемент \\"; }
            else { TB2.Text += TB.Text[i]; }
        }
    } else if (type == "int")
    { try { Convert.ToInt32(TB.Text);
            TB2.Text = TB.Text; }
        catch (FormatException)
        { L.Text = "Текст не является числом";
            flag = false;
            return false; }
    } if (flag) L.Text = "Ошибок не обнаружено";
    return true; }
```

Рис. 3. Листинг кода проверки и обработки введенных данных

Заключение

SQL-инъекции приводят к проблемам в работе веб-сайтов. Важно не допускать несанкционированного доступа к конфиденциальным данным, корректно обрабатывая входящую информацию. Закрытие уязвимостей от SQL-инъекций является важной задачей по обеспечению информационной безопасности и корректной работе веб-приложений. Данные проблемы решаемы с помощью инструментов программирования, пример решения рассмотрен в статье.

Список литературы

1. *Jones M.* Fight against SQL injection attacks [Электронный ресурс] // IBM. Режим доступа: https://www.ibm.com/developerworks/security/library/se-sql-injection-attacks/index.html.

2. *Егоров М.* Выявление и эксплуатация SQL-инъекций в приложениях // Защита информации. INSIDE. 2011. № 2. С. 2–8.

3. *Евтеев Д.* SQL Injection от А до Я [Электронный ресурс] // Режим доступа: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/PT-devteev-Advanced-SQL-Injection.pdf.

4. *Бирюков А. А.* Информационная безопасность: защита и нападение. М. : ДМК Пресс, 2012. 474 с.